LISTING OF CLAIMS

1. (**Currently Amended**) A computer-implemented method comprising:

automatically detecting a change introduced into a software object of a first software subsystem, where the first software subsystem stores the software object from which instances of the software object are created, wherein the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem**, where the first software subsystem is located at a server and the second software subsystem is located at a client**;

accessing a compatible changes database in response to detecting the change, where the compatible changes database indicates changes defined to be compatible with the software objects of the second **software** subsystem;

determining, based on accessing the compatible changes database, whether the change is compatible with the software objects of the second software subsystem, including determining whether the change is predefined as compatible; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

2-3. (Canceled)

4. (Previously Presented) The method of claim 1 further comprising issuing a message that the change is not allowed if the change is not predefined as compatible.

5. (Original) The method of claim 4 further comprising allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

6. (**Currently Amended**) A computer-implemented method comprising:

identifying a subset of software objects of a first software subsystem that are stored in the first software subsystem and are to be instantiated in a second software subsystem to interact with software objects of the second software subsystem, and declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems**, where the first software subsystem is located at a server and the second software subsystem is located at a client**;

detecting a change introduced into a frozen software object from the subset of software objects; and prior to allowing the change,

accessing a compatible changes database in response to detecting the change, where the compatible changes database indicates changes predefined to be compatible with the software objects of the second **software** subsystem;

determining with a compatibility check, based on accessing the compatible changes database, whether the change is compatible with a second software subsystem, including determining whether the change is predefined as compatible; and

issuing a notice indicating results of the compatibility check.

7. (Canceled)

8. (Previously Presented) The method of claim 6 wherein frozen software objects are classified to include released objects and restricted objects, where released objects are defined as software objects having a number of instances instantiated in the second software subsystem exceeds a threshold number, and restricted objects are defined as software objects having a number of instances instantiated in the second software subsystem that does not exceed the threshold number.

9-10. (Canceled)

11. (Previously Presented) The method of claim 8 wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the restricted object.

12. (Previously Presented) The method of claim 8 wherein classification of each frozen software object is based on a number of instances of each frozen software object occurring in the second software subsystem.

13. (Original) The method of claim 6 wherein a software object is a function module.

14. (Original) The method of claim 6 wherein a software object is a data structure.

15. (Original) The method of claim 13 wherein the software object includes an environment of the function module.

16. (Original) The method of claim 6 wherein a software object includes a class and an environment of the class.

17. (Original) The method of claim 6 wherein a software object includes an interface and an environment of the interface.

18. (Original) The method of claim 6 wherein a software object includes a program and an environment of the program.

19. (Original) The method of claim 6 wherein the detecting the change comprises automatically monitoring development of software code.

20. (Canceled)

21. **(Currently Amended)** The method of ~~**claim 7**~~ **claim 6** wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

22. **(Currently Amended)** A computer-implemented method comprising:

monitoring a software development space to detect changes **[[in]]** introduced into a subset of frozen software objects stored in a first software ~~**system**~~ **subsystem** that are to be instantiated in a second software subsystem, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems**, where the first software subsystem is located at a server and the second software subsystem is located at a client**;

performing a global compatibility check of the subset of frozen software objects of the first software subsystem by determining whether any changes were introduced into **[[a]]** the subset of frozen software objects of the first software subsystem since the time of a last compatibility check wherein the introduced changes were unapproved changes introduced without obtaining prior approval, where performing the global compatibility check includes comparing a current version of software code of a frozen **software** object in the software development space with a version of the software code of the frozen **software** object at the time of a last global compatibility check;

identifying software objects of the second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software ~~**system**~~ **subsystem**; and

issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change.

23-24. (Canceled)

25. (Original) The method of claim 24 wherein the frozen software objects include software objects of the first software subsystem used by software objects of the second software subsystem.

26-31. (Canceled)

32. (**Currently Amended**) An article of manufacture comprising:

a storage medium having stored therein instructions which, when executed by a processor, cause a processing system to perform a method comprising;

detecting a change introduced into a software object of a first software subsystem, where the first software subsystem stores the software object from which instances of the software object are created, wherein the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem**, where the first software subsystem is located at a server and the second software subsystem is located at a client**;

accessing a compatible changes database in response to detecting the change, where the compatible changes database indicates changes defined to be compatible with the software objects of the second **software** subsystem;

determining, based on accessing the compatible changes database, whether the change is compatible with the software objects of the second software subsystem, including determining whether the change is predefined as compatible; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

33. (Canceled)

34. (Previously Presented) The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method

further comprising issuing a notification that the change is not allowed if the change is not predefined as compatible.

35. (Previously Presented) The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method further comprising allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

36-38. (Canceled)